Remarks

Claims 1-52 are pending in this application. The examiner has objected to claims 2 and 4 due to a typographical error in the language of the claims. The examiner has rejected each of claims 1-52 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the applicants regard as their invention. The examiner has rejected claims 1, 19, 36, 40, and 46 under 35 U.S.C. § 112, second paragraph, as being incomplete for omitting essential steps. The examiner has rejected claims 36-39 under 35 U.S.C. § 101 for being directed to non-statutory subject matter. The examiner has rejected claims 1-52 under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,872,973 to Mitchell. The examiner has rejected claims 1, 19, 36, and 40 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 5,872,973 to Matsuda et al. Several of the claims of this application have been amended herein. For the convenience of the examiner a set of claims that includes all of the amendments made herein is attached as Exhibit A.

I. Objections, Section 112 Rejections, and Section 101 Rejections

The examiner has asserted objections to the abstract and the claims, rejected each of the claims under 35 U.S.C. § 112, and rejected claims 36-39 under 35 U.S.C. § 101 as being directed to non-statutory subject matter.

A. Objections to the Abstract

The examiner has objected to the abstract because of the use of words with initial capital letters. The abstract has been amended such that it does not include words with initial capital letters.

B. Objections to the Claims

The examiner has objected to claims 2 and 4 due to typographical errors in the claims. These claims have been corrected to remove the typographical errors.

C. Section 112 Rejections

The examiner has rejected claims 1, 4, 8, 11, 30, 36, 40, 41, and 46 as being indefinite due to the use of the phrase "adapted to." These claims have been amended to delete the use of the phrase "adapted to" and include a positive limitation in its place.

The examiner has rejected claim 1 as being indefinite due to the use of the phrase "the first Behavior logic." Claim 1 has been amended to delete the phrase "the first Behavior logic."

The examiner has rejected claims 40 and 41 as being indefinite due to the use of the terms "shared environment" and "first Behavior logic." Claims 40 and 41 have been amended to clarify the use of the phrase "shared environment" and to delete the phrase "first Behavior logic."

The examiner has rejected claims 1, 19, 40, and 46 under 35 U.S.C. § 112, second paragraph, as being incomplete for omitting alleged essential steps, including the steps of command mapping and behavior, and creating a shared environment. Each of claims 1, 19, 40, and 46 has been amended to clarify that the mapping of a command to a behavior logic is accomplished on the basis of at least one characteristic of the command. A characteristic of the command could include the identity of the command or the content of the command.

With respect to the step of creating a shared environment, applicants submit that claims 1 and 40 do include a recitation of "code to create a shared environment." Similarly, claim 19 includes the step of "creating a shared environment," and claim 46 includes the recitation of "a plurality of shared environments." Applicants submit that, in view of the amendments herein and the positive recitation of elements already present in the claims, the rejection of claims 1, 19, 40, and 46, due to the alleged omission in these claims of an essential element, should be withdrawn.

The examiner has rejected claim 36 under 35 U.S.C. § 112, second paragraph, as being incomplete for omitting an alleged essential step, including the step of configuring a command-receiver behavior logic. The step of configuring a receiver is recited in amended claim 36, namely: "configuring a receiver logic to receive a command and initiate the execution of at least one behavior logic corresponding to the command in response to the mapping of the command to one or more behavior logics." Because the configuring step is present in claim 36, applicants submit that the rejection of claim 36 under 35 U.S.C. § 112 for the omission of an essential step should be withdrawn.

D. Section 101 Rejections

The examiner has rejected claims 36-39 as being directed to non-statutory subject matter. Independent claim 36 has been amended to clarify that the application of claim 36 is "for execution on at least one computer system." Claims 37-39 depend from claim 36. Applicants submit that the rejection of claims 36-39 under 35 U.S.C. § 101 should be withdrawn.

II. Section 102 Rejections

The examiner has rejected each of the pending claims 1-52 as being anticipated by one or both of Mitchell and Matsuda.

A. Standard for Rejection Under 35 U.S.C. § 102

"A claim is not anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631 (Fed. Cir. 1987). "The identical invention must be shown in complete detail as is contained in the . . . claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1336 (Fed. Cir 1989). Thus, a rejection of any one of the pending claims is not permissible unless each element of the rejected claim is present in either Mitchell or Matsuda. Here, neither Mitchell nor Matsuda discloses each element of any of the rejected claims.

B. Mitchell Does Not Disclose the Claimed Invention

A plain reading of Mitchell demonstrates that Mitchell does not disclose the claimed invention. The claimed invention of the present application is directed to an object-oriented programming environment that includes a shared, distributed programming environment in which each object of the environment is able to map commands directed to the object to one of several behaviors of the object on the basis of a mapping function that is included within the object. The placement of the mapping functionality of the invention within the object allows the object to be autonomous and allows the object to function without regard to the location of the object within the programming environment. In addition, because the mapping functionality of the object is intrinsic to and within the object, an object can function in the programming environment without the necessity of defining relationships between the object and other objects of the programming environment.

In direct contrast, Mitchell is directed to a programming environment in which a "semantic link" — also referred to in Mitchell as a "surrogate object" or "mapper" — is placed *between* two other objects and serves to communicate with each of the objects. Mitchell provides that:

> [A] semantic link in the present invention is a system that synchronizes two server objects by creating a generic client between the servers.
>
> . . .
>
> The main function of the invention is to specify and then instantiate semantic links between objects that have been defined in a dynamic object-oriented language.
>
> . . .
>
> A semantic link is created through the instantiation of a surrogate object, called a mapper, that uses probing and dynamic binding to attach to both of its patron objects. The mapper is the client and both patron objects are servers to the mapper.
>
> . . .
>
> The objects are synchronized (that is a "message" is passed from one object to the other) through the mapper when the attribute changes its value, and the probes are called back.
>
> . . .
>
> In other words, two objects can be attached to each other by a third object without the first two objects becoming clients, thus creating a server/server architecture between the two objects.

(Mitchell, column 7, lines 36-38, 46-48, and 58-62; and column, 8, lines 18-21 and 32-35). The placement of a mapping function between two objects is also plainly shown in the figures of Mitchell. Figure 1 of Mitchell, for example, depicts a semantic link or mapper in the form of EosMapFieldToField object 103, which is positioned communicatively between two other objects, which are identified as patron object 102 and patron object 106. It is plain from Mitchell that Mitchell only discloses a mapping function placed between — *not within* — two objects.

Mitchell does not disclose the claimed invention of a shared, object-oriented environment in which the mapping of commands to behaviors for an object is accomplished according to a mapping functionality that resides within the object itself. The mapping function of the claimed invention is within each object, thereby allowing each object to function as an autonomous unit such that the object can be moved within the computer systems of the shared environment and function independently of its location in the shared environment. In addition, because the mapping function of each object is internal and intrinsic to each object, each object can be used without the necessity of defining relationships between the object and other objects in the programming environment. Each claim of the pending application includes the recitation of claim elements that are not disclosed or suggested by Mitchell.

1. *Independent Claim 1*

Claim 1 is directed to a storage medium containing software for manipulating computer-implemented objects in a computer system. Claim 1 includes the recitation of code to create an object. The claimed object of claim 1 includes "mapping logic able to map a command received at the receiver logic, on the basis of a characteristic of the command, to a selected behavior logic for execution of the selected behavior logic." This element of the claimed object of claim 1 is simply not present in Mitchell, as Mitchell is directed to the placement of the mapping function *between* two objects. In contrast, the mapping function of the object of claim 1 is within the object itself.

The examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that is being described in this section is the mapping of one object to another object through an external mapping function. The "mapping dialog" that is discussed in this section is a "dialog box" for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between "ObjectA" and "ObjectD."

In plain contrast, the object of the present invention includes a mapping function within the object itself. The placement of the mapping function within the object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 1 over Mitchell should be withdrawn.

2. *Independent Claim 19*

Independent claim 19 is directed to a method for manipulating a computer-implemented object in a distributed system. The step of "creating a plurality of objects" of claim 19 includes the steps of:

> selecting a behavior logic of the set of behavior logics corresponding to the command on the basis of a mapping logic within the object that maps commands to behavior logics of the set of behavior logics on the basis of a characteristic of the command;

(Claim 19). This step is not disclosed in Mitchell, as Mitchell is directed to the placement of the mapping function *between* two objects. In direct contrast with Mitchell, the mapping function of the claimed object is "within the object," thereby allowing the object to be autonomous and operable without regard to its location.

As discussed with respect to claim 1 above, the examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that is being described in this section is the mapping of one object to another object through an external mapping function. The "mapping dialog" that is discussed in this section is a "dialog box" for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between "ObjectA" and "ObjectD."

In plain contrast to this passage of Mitchell, the object of claim 19 includes a mapping function within the object itself. The placement of the mapping function within the object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 19 over Mitchell should be withdrawn.

3. *Independent Claim 36*

Independent claim 36 is directed to a method for designing an application for execution on at least one computer system. A step of "manipulating the object" of claim 36 includes the step of:

> mapping members of a first set of commands to members of the set of behavior logics, wherein the mapping function of an object is included within the object;
>
> . . .

> configuring a receiver logic to receive a command and initiate the execution of a behavior logic corresponding to the command in response to the mapping of the command to the behavior logic.

(Claim 36). These steps are not disclosed in Mitchell, as Mitchell is directed to the placement of the mapping function *between* two objects. In direct contrast with Mitchell, the mapping function of the claimed object is included "within the object," thereby allowing the object to be autonomous and operable without regard to its location and without the necessity of defining relationships between the object and other objects.

As discussed with respect to claims 1 and 19 above, the examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that is being described in this section is the mapping of one object to another object through an external mapping function. The "mapping dialog" that is discussed in this section is a "dialog box" for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between "ObjectA" and "ObjectD."

In plain contrast to this passage of Mitchell, the objects of claim 36 include a mapping function with each object itself. The placement of the mapping function within each object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 36 over Mitchell should be withdrawn.

4. *Independent Claim 40*

Independent claim 40 is directed to a processor-based system. The system includes code to create an object, and the created object includes:

> a mapping logic able to map a command received at the receiver logic to a selected behavior logic for execution of the selected behavior logic on the basis of a characteristic of the command.

(Claim 40). An object for use in a shared environment that includes a set of behavior logics and a mapping logic within the object is not disclosed in Mitchell. Mitchell, in contrast, is directed to the placement of the mapping function *between* two objects. In direct contrast with Mitchell, the mapping function of the claimed object of claim 40 is included within the object, thereby allowing the object to be autonomous and operable without regard to its location and without the necessity of defining relationships between the object and other objects.

As discussed with respect to claim 1, 19, and 36 above, the examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that is being described in this section is the mapping of one object to another object through an external mapping function. The "mapping dialog" that is discussed in this section is a "dialog box" for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between "ObjectA" and "ObjectD."

In plain contrast to this passage of Mitchell, the object of claim 40 includes a mapping logic within the object itself. The placement of the mapping logic within the object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 40 over Mitchell should be withdrawn.

5. *Independent Claim 46*

Independent claim 46 is directed to a software architecture. Claim 46 includes the recitation of a distributed system that includes "a plurality of shared environments." There is no disclosure in Mitchell of multiple shared environments. For the disclosure of a software architecture that includes multiple shared environments, the examiner points to three lines in column 6, two lines in the abstract, and Figures 1-3 of Mitchell. None of these passages or the figures discloses multiple shared environments. Instead, these passages simply disclose linking of objects and the external mapping of objects. There is not even a suggestion in any of these passages of a software architecture that includes *multiple* shared environments.

In addition, claim 46 includes "a Kernel subclass of the CommandReceiver class" that includes "code to instantiate objects of the CommandReceiver class" and "code to destroy objects of the CommandReceiver class." Neither of these code elements is disclosed in Mitchell. First, with respect to the element of "code to instantiate objects of the CommandReceiver class," the examiner has pointed to column 29, lines 55-67 of Mitchell. This passage of Mitchell says nothing about the instantiation of objects. Rather, this passage speaks again to links between objects. There is absolutely no mention in this passage of code for instantiating, or creating, an object of the CommandReceiver class.

In addition, the claim element of a Kernel subclass that includes "code to destroy objects of the CommandReceiver class," is not disclosed in Mitchell. For this element, the examiner points to the "garbage collection" discussion in column 17, lines 29-48 of Mitchell. There is no disclosure in this passage of code within a Kernel subclass of the CommandReceiver class for destroying objects of the CommandReceiver class. Code of this sort is simply not disclosed in Mitchell. Applicants respectfully submit that the rejection of claim 46 should be withdrawn.

In sum, each of the pending claims includes claim elements that are not present in Mitchell. Because Mitchell does not disclose each element of the claimed invention, a rejection of the pending claims under 35 U.S.C. § 102 over Mitchell is improper. The rejection of the pending claims on the basis of Mitchell should be withdrawn.

C. Matsuda Does Not Disclose the Claimed Invention

The examiner has rejected each of claims 1, 19, 36, and 40 as being anticipated by Matsuda. A plain reading of Matsuda reveals, however, that Matsuda does not anticipate any of the pending claims.

1. *Object-Oriented Programming Environment*

Each of claims 1, 19, 36, and 40 includes the recitation of the use or implementation of the invention in an object-oriented programming environment. Matsuda is not in any manner directed to an object-oriented programming environment. Instead, Matsuda is directed to a method for manipulating a graphical depiction or cartoon of a "pet." Although "the pet" of Matsuda is sometimes referred to in Matsuda as an "object," it is plain from a reading of

Matsuda that Matsuda does not concern object-oriented programming. The term "object-oriented programming" does not once appear in Matsuda. Moreover, the abstract of Matsuda makes clear that the terms "pet" and "object" are used synonymously in Matsuda, and are used to denote the graphical depiction or cartoon of a figure and not as shorthand for object-oriented programming. The "object" of Matsuda is not an object of an object-oriented programming environment. Applicants submit that the rejection of claims 1, 19, 36, and 40 should be withdrawn, as Matsuda does not disclose an object-oriented programming environment.

2. *An Object That Includes Behavior Logics*

Each of claims 1, 19, 36, and 40 includes the recitation of an object that includes a set of behavior logics. The pet of Matsuda does not include a set of behavior logics. The pet is simply a depiction or cartoon of an animal on a viewable medium. Matsuda describes the use of scripts to be executed by a client PC to cause the pet to move. (Matsuda, Paragraph 188). Even assuming, for the sake of argument, that these scripts are behavior logics, these scripts are not included in the pet. Instead, they exist in AO server 13 of Matsuda and are transmitted from AO server 13 of Matsuda for execution by a client PC. (Matsuda, Paragraph 188). The scripts of Matsuda are not included in the pet of Matsuda. As such, it cannot be said the Matsuda discloses behavior logics that are included within the object (pet) of Matsuda. Because Matsuda does not disclose an object that includes behavior logics, the rejection on claims 1, 19, 36, and 40 should be withdrawn.

3. *An Object That Includes A Mapping Logic*

Each of claims 1, 19, 36, and 40 includes the recitation of an object that includes mapping logic or mapping functionality for mapping commands received by the logic to behavior logics of the object. The pet of Matsuda does not include a mapping logic. As described above, the pet is simply a depiction of an animal on a viewable medium. Matsuda describes that commands from a user are interpreted with the assistance of a behavior command control table in AO server 13 of Matsuda. (Matsuda, Paragraph 185). Even assuming, for the sake of argument, that the behavior command control table of Matsuda is a mapping logic, this table is not included in the pet. Instead, the table exists in the AO server. The table and the AO server of Matsuda are not included within the "pet" of Matsuda. Because the table is not included within the pet, it cannot be argued that Matsuda discloses a mapping logic that is included within the object (pet) of Matsuda. Because Matsuda does not disclose an object that includes a mapping logic, the rejection on claims 1, 19, 36, and 40 as being anticipated by Matsuda should be withdrawn.

In summary, each of claims 1, 16, 36, and 39 includes claim elements that are not disclosed by Matsuda. Because Mitchell does not disclose each element of the claimed invention, a rejection of the pending claims under 35 U.S.C. § 102 over Mitchell is improper. Applicants respectfully submit that the rejection of these claims should be withdrawn, and these claims should be passed to issuance.

D. Dependent Claims 2-18, 20-35, 37-39, 41-45, and 47-52

Dependent claims 2-18, 20-35, 37-39, 41-45, and 47-52 will not be discussed individually herein, as each depends from an otherwise allowable base claim. Applicants submit that the rejection of claims 2-18, 20-35, 37-39, 41-45, and 47-52 should be withdrawn and these claims should be passed to issuance.

Conclusion

Applicants respectfully submit that the rejection of claims 1-52 should be withdrawn, and these claims should be passed to issuance.

Respectfully submitted,

[illegible]

Roger Fulghum
Registration No. 39,678

Baker Botts L.L.P.
910 Louisiana
One Shell Plaza
Houston, Texas 77002-4995
(713) 229-1707

Baker Botts Docket Number: 002376.0926

Date: January 5, 2005

Exhibit A

Exhibit A

Amendments to the Claims

A complete list of pending claims follows, with all amendments entered:

1. (Currently Amended) A storage medium containing software for manipulating computer-implemented objects in a distributed system, the software comprising:

code to create a shared environment, the shared environment comprising an object-oriented programming environment distributed across multiple computer systems and comprising a plurality of objects; and

code to create an object, the object exposed to other objects in the shared environment, the object comprising:

a set of behavior logics, each member of the set of behavior logics operable to cause the object to perform a task;

a receiver logic operable to receive a command from another object in the shared environment, wherein the receiver logic is externally invokable; and

a mapping logic able to map a command received at the receiver logic, on the basis of a characteristic of the command, to a selected behavior logic for execution of the selected behavior logic.

2. (Currently Amended) The storage medium of claim 1, wherein the set of behavior logics and the mapping logic are private to the object.

3. (Currently Amended) The storage medium of claim 1, wherein the set of behavior logics has no members.

Exhibit A

4. (Currently Amended) The storage medium of claim 1, the object further comprising:

a default behavior logic operable to cause the object to perform a default task, wherein the default behavior logic is private to the object and wherein the default behavior logic is executed if the received command is not mapped to another behavior logic.

5. (Currently Amended) The storage medium of claim 1, wherein a command can be mapped to multiple behavior logics.

6. (Currently Amended) The storage medium of claim 1, the object further comprising:

an authentication data, the authentication data providable to other objects for authenticating commands received from the other objects by the code to receive the command.

7. (Currently Amended) The storage medium of claim 6, wherein the command comprises the authentication data, and wherein the mapping of a command to a behavior logic may be restricted in response to the authentication data.

8. (Currently Amended) The storage medium of claim 1, the software further comprising:

code to create a first Shadow of the object, the first Shadow of the object operable to communicate with the object, the first Shadow of the object being informed of changes to the object and the object being informed of changes to the first Shadow of the object.

9. (Original) The storage medium of claim 8, wherein the first Shadow of the object is a copy of the object.

10. (Currently Amended) The storage medium of claim 8, wherein the mapping of commands to behavior logics of the first Shadow of the object differs from the mapping of commands to behavior logics of the object.

11. (Currently Amended) The storage medium of claim 8, the software further comprising:

code to create a plurality of Shadows of the object operable to communicate with the object and the first Shadow of the object, the object and the first Shadow of the object being informed of changes to any of the plurality of Shadows of the object and each of the plurality of Shadows of the object being informed of changes to the object and changes to the first Shadow of the object.

12. (Original) The storage medium of claim 8, the software further comprising:

code to promote the first Shadow of the object into a new object.

13. (Original) The storage medium of claim 12, the software further comprising:

code to create a plurality of Shadows of the object,

wherein executing the code to promote the first Shadow of the object into a new object converts each of the plurality of Shadows of the object into a Shadow of the new object.

14. (Currently Amended) The storage medium of claim 12, the shared environment further comprising:

a plurality of computer systems, the object on a first computer system of the plurality of computer systems, the first Shadow of the object on a second computer system of the plurality of computer systems; and

code to manage the plurality of computer systems, executing the code to promote the first Shadow of the object to a new object if the first computer system experiences a predetermined condition.

15. (Currently Amended) The storage medium of claim 1, the set of Behavior logics further comprising:

code to modify the mapping logic to modify the mapping of commands to behavior logics.

16. (Currently Amended) The storage medium of claim 1,

wherein the shared environment is operable to execute on a plurality of computer systems; and

wherein the object has a location on one of the plurality of computer systems and wherein the object acts independently of its location.

Exhibit A

17. (Currently Amended) The storage medium of claim 1, the object further comprising:

code to configure the mapping logic from an external data source.

18. (Currently Amended) The storage medium of claim 1, wherein the software is capable of using any networking protocol.

19. (Currently Amended) A method of manipulating a computer-implemented object in a distributed system, the method comprising the steps of:

creating a shared environment, the shared environment comprising an object-oriented programming environment distributed across multiple computer systems and comprising a plurality of objects; and

creating an object, wherein the object is exposed to other objects in the shared environment, and wherein the step of creating an object comprises the step of:

coding a set of behavior logics, each member of the set of behavior logics causing the object to perform a task;

manipulating the object, wherein the step of manipulating the object comprises the steps of:

receiving a command from another object of the plurality of objects in the shared environment;

selecting a behavior logic of the set of behavior logics corresponding to the command on the basis of a mapping logic within the object that maps commands to behavior logics of the set of behavior logics on the basis of a characteristic of the command; and

executing the selected behavior logic responsive to the command.

20. (Currently Amended) The method of claim 19, wherein the set of behavior logics and the mapping logic are private to the object.

21. (Currently Amended) The method of claim 19, further comprising the step of:

modifying the mapping logic to modify the mapping of commands to behavior logics.

22. (Currently Amended) The method of claim 19, the method further comprising the steps of:

coding a default behavior logic to cause the object to perform a default task; and

executing the default behavior logic if no other behavior logic from the set of behavior logics is mapped to the received command.

23. (Currently Amended) The method of claim 19, wherein the set of behavior logics has no members.

24. (Currently Amended) The method of claim 19, wherein multiple behavior logics are mapped to and selected on the basis of a received command.

Exhibit A

25. (Currently Amended) The method of claim 19, further comprising the step of:

creating an authentication data for the object.

26. (Currently Amended) The method of claim 25, the command comprising the authentication data, the method further comprising the step of:

restricting the mapping of commands to behavior logics in response to the authentication data.

27. (Currently Amended) The method of claim 19, further comprising the step of:

creating a first Shadow of the object, the first Shadow of the object operable to communicate with the object, the first Shadow of the object being informed of changes to the object and the object being informed of changes to the first Shadow of the object.

28. (Original) The method of claim 27, the step of creating the first Shadow of the object comprising the step of:

copying the object.

29. (Currently Amended) The method of claim 27, the step of creating the first Shadow of the object comprising the step of:

modifying the mapping of commands to behavior logics of the first Shadow of the object.

Exhibit A

30. (Currently Amended) The method of claim 27, further comprising the step of:

creating a plurality of Shadows of the object operable to communicate with the object and the first Shadow of the object, the object and the first Shadow of the object being informed of changes to any of the plurality of Shadows of the object and each of the plurality of Shadows of the object being informed of changes to the object and changes to the first Shadow of the object.

31. (Original) The method of claim 27, further comprising the step of:

promoting the first Shadow of the object into a new object.

32. (Original) The method of claim 31, further comprising the step of:

creating a plurality of Shadows of the object,

converting each of the plurality of Shadows of the object into a Shadow of the new object, responsive to the step of promoting the first Shadow of the object.

33. (Currently Amended) The method of claim 19,

wherein the object has a location on a first computer system of the plurality of computer systems and wherein the object acts independently of its location.

34. (Original) The method of claim 19, the shared environment capable of using any networking protocol to communicate with another shared environment.

35. (Currently Amended) The method of claim 19, further comprising the step of:

creating the mapping logic from an external data source.

36. (Currently Amended) A method for developing an application for execution on at least one computer system from configurable objects having behavior logics capable of performing tasks, the method comprising the steps of:

defining within an object-oriented programming environment a plurality of objects, each object of the plurality of objects operable to receive and execute commands, each object exposed to each other object of the plurality of objects, the step of creating the plurality of objects comprising the steps of:

creating a set of behavior logics for an object;

mapping members of a first set of commands to members of the set of behavior logics, wherein the mapping function of an object is included within the object;

mapping any command not a member of the first set of commands to a default behavior logic; and

configuring a receiver logic to receive a command and initiate the execution of a behavior logic corresponding to the command in response to the mapping of the command to the behavior logic.

37. (Currently Amended) The method of claim 36, further comprising the steps of:

creating a Shadow of an object of the plurality of objects, the Shadow configured such that sending a command to the Shadow causes the object to act as if the command had been sent to the object.

Exhibit A

38. (Currently Amended) The method of claim 37, each of the plurality of objects having a location on one of a plurality of computer systems, each of the plurality of objects being independent of the location of each other of the plurality of objects.

39. (Currently Amended) The method of claim 38, wherein a Shadow of each of the plurality of objects is automatically created on each of the plurality of servers other than the server on which the object is located.

40. (Currently Amended) A processor-based system, comprising:

a first processor; and

a first storage device coupled to the first processor containing software to manipulate computer-implemented objects in a shared environment, the software comprising:

code to create a shared environment, the shared environment comprising an object-oriented programming environment distributed across multiple computer systems and comprising a plurality of objects; and

code to create an object of the plurality of objects, the object exposed to other objects in the shared environment, the object comprising:

a set of behavior logics, each member of the set of behavior logics operable to cause the object to perform a task; and

a receiver logic operable to receive a command from another object in the shared environment, the receiver logic being externally invokable;

a mapping logic able to map a command received at the receiver logic to a selected behavior logic for execution of the selected behavior logic on the basis of a characteristic of the command.

41. (Currently Amended) The processor-based system of claim 40, the object further comprising:

a default behavior logic operable to cause the object to perform a default task, wherein the default Behavior logic is private to the object and wherein the default behavior logic is executed if the received command is not mapped to another behavior logic.

42. (Currently Amended) The processor-based system of claim 40, wherein a command can be mapped to multiple behavior logics.

43. (Currently Amended) The processor-based system of claim 40, further comprising:

an input device coupled to the first processor,

wherein a first object of the plurality of objects is coupled to the input device such that manipulation of the input device sends a command from the first object to a second object of the plurality of objects without identifying the input device, the second object of the plurality of objects acting responsive to the command independent of the nature of the input device.

44. (Original) The processor-based system of claim 40, further comprising:

an output device coupled to the first processor, wherein a first object of the plurality of objects is coupled to the input device such that a first object is capable of rendering a second object on the output device without identifying the output device to the second object.

45. (Original) The processor-based system of claim 40, further comprising:

a second processor;

a network, coupled to the first processor and the second processor;

a second storage device coupled to the second processor, the second storage device containing the software;

the software further comprising:

code to connect the shared environment to the network;

code to create a Shadow on the second processor of the object on the first processor, the Shadow and the object communicating with each other to inform the Shadow of changes to the object and the object of changes to the Shadow.

46. (Currently Amended) A software architecture for manipulating computer-implemented objects on a plurality of computers, some of the plurality of computers having input devices and some of the plurality of computers having output devices, the software architecture implemented in an extensible object-oriented language, comprising:

a distributed system, comprising:

a plurality of shared environments, each of the plurality of shared environments comprising an object-oriented programming environment distributed across and executing on a different computer of the plurality of computers, each of the plurality of shared environments comprising:

a CommandReceiver class, the CommandReceiver class comprising:

a set of Behavior private methods, each member of the set of Behavior methods operable to cause instantiations of the CommandReceiver class to perform a task; and

an executeCommand public method, operable to receive a Command from an object in the shared environment, the executeCommand public method comprising:

code to receive the Command;

code to select a Behavior private method of the set of Behavior private methods selected corresponding to a characteristic of the Command from a Command-Behavior mapping; and

code to execute the selected Behavior private method; and

a Kernel subclass of the CommandReceiver class, the Kernel class comprising:

code to instantiate objects of the CommandReceiver class;

code to destroy objects of the CommandReceiver class.

47. (Original) The software architecture of claim 46, further comprising:

a Pawn subclass of the CommandReceiver class, the Pawn subclass comprising:

code to register an instantiation of a Pawn with a Kernel object of the Kernel subclass;

code to determine whether an instantiation of the Pawn subclass is a real Pawn or a Shadow Pawn of a real Pawn, and

code to send State information about an instantiation of the Pawn subclass,

wherein Commands received by Shadow Pawns are sent to the real Pawn corresponding to the Shadow Pawn.

48. (Original) The software architecture of claim 46, further comprising:

a ControlDevice subclass of the CommandReceiver class corresponding to an input device for receiving input from the input device and sending Commands to other CommandReceiver objects.

49. (Original) The software architecture of claim 46, further comprising:

a Construct subclass of the CommandReceiver class corresponding to an output device for rendering objects of the CommandReceiver class with graphical attributes.

50. (Original) The software architecture of claim 46, further comprising:

a Console subclass of the CommandReceiver class for allowing a user of the distributed system to instantiate, modify, and destroy objects, and for allowing a user to send Commands to CommandReceiver objects.

51. (Original) The software architecture of claim 46, further comprising:

a Nengine subclass of the CommandReceiver class for serializing and deserializing CommandReceiver objects, transmitting and receiving the serialized CommandReceiver object across a network to a Nengine in another shared environment of the distributed system.

52. (Original) The software architecture of claim 51, further comprising:

a Node subclass of the CommandReceiver class, an instantiation of the Node subclass corresponding to a Pawn object for representing the Pawn object to a Nengine object for communicating State information corresponding to a Pawn to Shadow Pawns of the Pawn and for communicating Commands sent to a Shadow Pawn to the real Pawn corresponding to the Shadow Pawn.